STUDIES OF LEXICOGRAPHY IN THE GENERALIZED NETWORK SIMPLEX METHOD

Jose C. Arantes
Department of Mechanical, Industrial and
Nuclear Engineering
University of Cincinnati
Cincinnati, OH 45221-0072

John R. Birge
and
Katta G. Murty
Department of Industrial & Operations Engineering
University of Michigan
Ann Arbor, MI 48109-2117

Technical Report 92-12
January 1992

STUDIES OF LEXICOGRAPHY IN THE GENERALIZED NETWORK SIMPLEX METHOD

José C. ARANTES
Department of Mechanical, Industrial and Nuclear Engineering, University of Cincinnati, Cincinnati, Ohio, 45221-0072, USA

and

J. R. BIRGE and K. G. MURTY
Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, Michigan 48109-2117, USA

Abstract

This paper introduces an analytical approach for studying lexicography in generalized network problems. The equations obtained can help us to understand and extend the existing theory. First, it is verified that all nonzero elements have the same sign in each row vector of a basis inverse for a generalized network (GN) problem with positive multipliers. However, this property does not necessarily hold when there exists negative multipliers. Second, we developed a strategy to select the dropping arc in the GN simplex algorithm when addressing GN problems with positive and *negative* multipliers. This strategy is also based on lexicography and requires performing some comparisons. However, the values to be compared are already known since they can be obtained as a by-product of the calculations necessary to compute the basis representation of the entering arc. Consequently, the computational effort per pivot step is O(n) in the worst case. This worst case effort is the same as that required by the strongly convergent rules for selecting the dropping arc in the method of strong convergence.

Key words: linear programming, generalized networks, simplex method, degeneracy, lexicography, cycling

1. Introduction

Let $G = (\mathcal{N}, \mathcal{A})$ be a directed connected generalized network (GN) with $\mathcal{N}$ = set of nodes, $\mathcal{A}$ = set of arcs, where $|\mathcal{N}| = n$, $|\mathcal{A}| = m$ and $\mathcal{A}$ may contain self-loops at some or all the nodes. Unlike in pure networks, flows in GN are subject to a linear gain or loss as they travel through the arcs, i.e., if a flow of f_{ij} units enters an arc (i, j) at its tail node i, it becomes $p_{ij}f_{ij}$ units by the time it reaches the head node j; the factor p_{ij} is known as the multiplier associated with arc (i, j); and f_{ij} is known as the flow in arc (i, j) or the flow variable associated with it. The general problem, known as the generalized minimum cost flow problem (GMCF) on G is

$$\text{Minimize} \quad \sum_{(i,j)\in\mathcal{A}} c_{ij}f_{ij}$$

$$-\sum_{j:(i,j)\in\mathcal{A}} f_{ij} + \sum_{j:(j,i)\in\mathcal{A}} p_{ji}f_{ji} = b_i, \qquad i \in \mathcal{N} \tag{1}$$

$$0 \le f_{ij} \le u_{ij} \qquad (i, j) \in \mathcal{A}$$

The coefficients of f_{ij}, for $(i, j) \in \mathcal{A}$ and $i \neq j$, are -1 and p_{ij} respectively in the constraints in (1) corresponding to nodes i and j. The coefficient of f_{ii}, for the self-loop (i, i) $\in \mathcal{A}$, appears in only the constraint in (1) corresponding to node i, we assume that its coefficient there is scaled to be either -1 or +1; the self loop is said to be slack (surplus) self-loop depending on whether this coefficient is +1 (or -1). The multiplier associated with any self-loop is by definition +1. In (1), b_i is the requirement at node i. The variables f_{ii} in (1) associated with self-loops in G, represent slack or surplus variables associated with the requirement constraint at $i \in \mathcal{N}$.

Currently, variants of the primal simplex algorithm implemented using the rooted loop labeling data structures to represent the quasitrees in the basis network [2, 4, 6] are among the most efficient practical algorithms for solving the GMCF. Similarly to pure network flow problems, GN problems tend to be usually degenerate. The strongly convergent (SC) primal simplex algorithm of Elam, Glover and Klingman [2] resolves the problem of cycling under degeneracy in the GMCF. However, as pointed out by them, the SC algorithm works only when all the multipliers are positive. The equivalence between strong convergence and lexicography has been established by Partovi [8] and Orlin [7]. In this paper, we develop an efficient implementation of lexicography for solving GMCF with multipliers of arbitrary signs. This implementation requires, in each pivot step, the same worst case effort as the SC rules.

The following section presents some results on the the basis inverse. Section 3 introduces the ratio matrix R and presents an approach, based on R and lexicography, for selecting the dropping arc in GN with multipliers of arbitrary sign. This approach requires the same effort as the SC rules. Section 4 presents some concluding remarks.

2. Results on the Basis Inverse

Since (1) is a GMCF, there are no redundant equality constraints in (1), see [2, 4, 6]. A basic solution for (1) corresponds to a partition of arcs in $\mathcal{A}$ into (**Q**, **L**, **U**), where **Q** = set of basic arcs in this partition, satisfying |**Q**| = n, and the column vectors associated with arcs in **Q** forming a nonsingular square submatrix B of order n of the coefficient matrix A of (1). The nonbasic arcs are partitioned into **L**, **U** which are respectively the sets of nonbasic arcs on which the flow is equal to the lower bound and capacity respectively. B is known as the working basis associated with the partition (**Q**, **L**, **U**). The basic solution of (1) corresponding to the partition (**Q**, **L**, **U**) is obtained by fixing the flows on the arcs in **L** (**U**) at their lower bound (capacity) and then evaluating the flows on the remaining basic arcs so as to satisfy the equality constraints. The partition is said to be primal feasible if the flows on basic arcs satisfy the lower and upper bounds.

The basic arcs in **Q** in a partition (**Q**, **L**, **U**) form a spanning subgraph of G, with one or more connected components, each of which is a quasitree, i.e., a tree with one additional arc which may be a self-loop. So each quasitree contains a cycle which may be a self-loop. In rooted loop labeling, node labels are defined in each quasitree separately. If the cycle in a quasitree is a self-loop it is considered to be a forward (reverse) arc if it is a slack (surplus) arc, and it is the unique root node for the quasitree. If the cycle in a quasitree is not a self-loop, then an arbitrary predecessor direction is selected for it, and the in-cycle arcs are classified as forward or reverse accordingly. When the cycle arcs are deleted from a quasitree, it is obtained a node disjoint collection of trees called its tributary trees. Each tributary tree contains exactly one cycle node which is its root node. Arcs in the tributary trees are classified as forward arcs if they are directed away from the root node, reverse otherwise.

Let S be a subset of **Q**. FOR(S) and REV(S) denote the sets of forward and reverse arcs in S. The labelling of nodes is such that, if (i, j) $\in$ FOR(**Q**) then i is a *predecessor* of j; i is a *successor* of j otherwise. The *collective gain/loss factor* of S, $\psi(S)$, is defined to be

$$\psi(S) = \frac{\prod_{(i,j) \in REV(S)} p_{ij}}{\prod_{(i,j) \in FOR(S)} p_{ij}}$$

where $\prod_{(i,j) \in S} p_{ij}$ is defined to be +1 if S = $\emptyset$.

The predecessor path of a node g in a quasitree, denoted by $\mathcal{P}_g$ (the same symbol will also be used to denote the set of nodes or the set of arcs on it) is obtained by a trace of the predecessor indices beginning with g, and terminates when a node repeats. So $\mathcal{P}_g$ repeats no arcs, and always contains the cycle or self-loop in the quasitree containing g. The symbol $\mathcal{P}_{ig}$ denotes the portion of the predecessor path of g up to node i if i is an ancestor of g, or the empty path otherwise.

Let B be the basis associated with a partition (**Q**, **L**, **U**). Since rows, columns of B are associated with nodes in $\mathbf{N}$, arcs in **Q** respectively; rows, columns of B^{-1} are associated with arcs in **Q**, nodes in $\mathbf{N}$ respectively. Let $\beta_{.}(k)$ = column of B^{-1} associated with node k, $\beta_{ij}(.)$ = row of B^{-1} associated with arc (i, j) $\in$ **Q** and $\beta_{ij}(k)$ = entry in B^{-1} in row associated with (i, j) $\in$ **Q** and column associated with node k. Then $B\beta_{.}(k) = I_{.k}$, the *k*th column vector of the unit matrix of order n; i.e., $\beta_{.}(k)$ is the vector of flow values on the basic arcs in **Q** to satisfy a requirement of one unit at node k and zero units at all other nodes, with zero flows on all nonbasic arcs. From this we see that $\beta_{ij}(k) \neq 0$ iff (i, j) $\in \mathcal{P}_k$. Actually, see [2, 6],

$$\beta_{ij}(k) = \delta\alpha\psi(\mathcal{P}_{ik}), \tag{2}$$

where $\delta = 0$ if $(i, j) \notin \mathcal{P}_k$, $+1$ if $(i, j) \in \text{FOR}(\mathcal{P}_k)$, or -1 if $(i, j) \in \text{REV}(\mathcal{P}_k)$; $\alpha = [1 - \psi(C_k)]^{-1}$ if (i, j) is on the non-self-loop cycle C_k in the quasitree containing node k, or 1 otherwise. Result 1 follows from (2).

Result 1: For $(i, j) \in \mathbf{Q}$ and $k \in \mathbf{N}$, $\beta_{ij}(k) = 0$ if $(i, j) \notin \mathcal{P}_k$; $\beta_{ij}(k) \neq 0$ otherwise.

Result 2: From the above, if p, q are any pair of nodes in the same quasitree containing an arc (i, j), then $\beta_{ij}(p)$ and $\beta_{ij}(q)$ are both nonzero iff $(i, j) \in \mathcal{P}_p \cap \mathcal{P}_q$, and

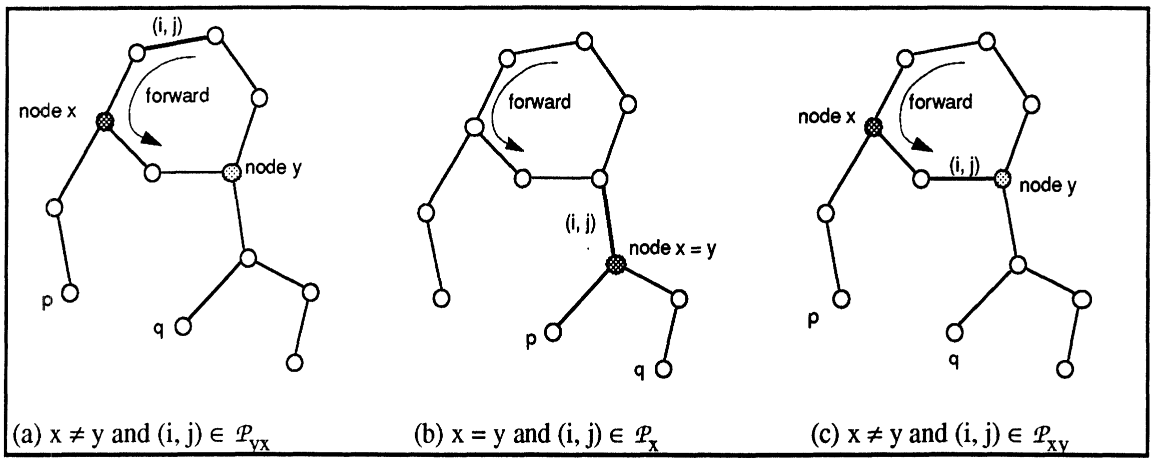
$$\beta_{ij}(p) = \frac{\psi(\mathcal{P}_{xp})}{\psi(\mathcal{P}_{xq})}\beta_{ij}(q) \qquad \text{if} \quad x \neq y \text{ and } (i, j) \in \mathcal{P}_{yx}$$

$$\qquad\qquad\qquad\qquad\qquad \text{or} \quad x = y \text{ and } (i, j) \in \mathcal{P}_{x}$$

$$\beta_{ij}(p) = \frac{\psi(\mathcal{P}_{yp})}{\psi(\mathcal{P}_{yq})}\beta_{ij}(q) \qquad \text{if} \quad x \neq y \text{ and } (i, j) \in \mathcal{P}_{xy}$$

where x = the node in $\mathcal{P}_p$ such that $\mathcal{P}_{xp} = \mathcal{P}_p \backslash \mathcal{P}_q$, y = the node in $\mathcal{P}_q$ such that $\mathcal{P}_{yq} = \mathcal{P}_q \backslash \mathcal{P}_p$. See figure 1.

Note that nodes x and y are only defined when nodes p and q belong to the same quasitree. These results follow because $\psi(\mathcal{P}_{ip}) = \psi(\mathcal{P}_{xp})\psi(\mathcal{P}_{ix})$ and $\psi(\mathcal{P}_{iq}) = \psi(\mathcal{P}_{xq})\psi(\mathcal{P}_{ix})$ if either $x = y$ and $(i, j) \in \mathcal{P}_x$ or if $x \neq y$ and $(i, j) \in \mathcal{P}_{yx}$; $\psi(\mathcal{P}_{ip}) = \psi(\mathcal{P}_{yp})\psi(\mathcal{P}_{iy})$ and $\psi(\mathcal{P}_{iq}) = \psi(\mathcal{P}_{yq})\psi(\mathcal{P}_{iy})$ if $x \neq y$ and $(i, j) \in \mathcal{P}_{xy}$.



(a) $x \neq y$ and $(i, j) \in \mathcal{P}_{yx}$ (b) $x = y$ and $(i, j) \in \mathcal{P}_{x}$ (c) $x \neq y$ and $(i, j) \in \mathcal{P}_{xy}$

Figure 1: Different relative positions of nodes p and q and $(i, j) \in \mathcal{P}_p \cap \mathcal{P}_q$.

Result 2 directly leads to the following result

Result 3: If $p_{ij} > 0$ for all $(i, j) \in \mathbf{Q}$ then all nonzero entries in each row of $\beta = B^{-1}$ have the same sign. This result may not hold if $p_{ij} < 0$ for some $(i, j) \in \mathbf{Q}$.

3. Results on the Ratio Matrix and the Lexicographic Rules

Let $(\mathbf{Q}, \mathbf{L}, \mathbf{U})$ be the feasible partition associated with the basis B, and (p, q) be the entering arc in some pivot step. Let $\bar{f} = (\bar{f}_{ij}\colon (i, j) \in \mathcal{A})$ be the basic feasible solution (BFS) associated with $(\mathbf{Q}, \mathbf{L}, \mathbf{U})$. In this section p, q always refer to nodes on the entering arc in the pivot step. Let $(\bar{a}_{ij}\colon (i, j) \in \mathbf{Q})$ be the basis representation of (p, q), i.e., $(\bar{a}_{ij})$ is the pivot column (or the updated column of the entering variable) in this pivot step. Let $\beta_{ij}(p)$, $\beta_{ij}(q)$ denote the entries in B^{-1} in the row associated with $(i, j) \in \mathbf{Q}$, and the columns associated with p, q. The value of $\bar{a}_{ij}$, for $(i, j) \in \mathbf{Q}$, can be obtained by:

$$\bar{a}_{ij} = \begin{cases} -\beta_{ij}(p) + p_{pq}\,\beta_{ij}(q) & \text{if } p \neq q \\ \beta_{ij}(p) & \text{if } p = q \text{ and } (p, q) \text{ is a slack self-loop} \\ -\beta_{ij}(p) & \text{if } p = q \text{ and } (p, q) \text{ is a surplus self-loop} \end{cases} \quad (3)$$

When it is primal feasible, the partition $(\mathbf{Q}, \mathbf{L}, \mathbf{U})$ is said to be lexico primal feasible, see [5], if $\beta_{ij}(.)$ is lexicopositive ($\succ 0$) for $(i, j) \in \mathbf{Q}$ satisfying $\bar{f}_{ij} = 0$ and $\beta_{ij}(.)$ is lexiconegative ($\prec 0$) for $(i, j) \in \mathbf{Q}$ satisfying $\bar{f}_{ij} = u_{ij}$. The bounded variable lexico simplex method [5] is initiated with a lexico primal feasible partition, and executed using a special dropping arc choice rule called the lexicographic rules, in each pivot step, to guarantee preservation of the lexico primal feasibility property. Let $\mathbf{D}$ denote the set of all the arcs in $\mathbf{Q} \cup \{(p, q)\}$ eligible to be the dropping arc in this pivot step, see [2, 4, 6]. If $\mathbf{D}$ is a singleton then the dropping arc is identified unambiguously. However, if $|\mathbf{D}| \geq 2$ then the lexicographic rules obtains the dropping arc (r, s) as follows:

Lexicographic Rules: The dropping arc in this pivot step is (p, q) iff either (p, q) $\in$ $\mathbf{L} \cap \mathbf{D}$ and $(\frac{1}{\bar{a}_{ij}})\, \beta_{ij}(.) \succ 0$ for all (i, j) $\in$ $\mathbf{D} \cap \mathbf{Q}$, or (p, q) $\in$ $\mathbf{U} \cap \mathbf{D}$ and $(\frac{1}{\bar{a}_{ij}})\, \beta_{ij}(.) \prec 0$ for all (i, j) $\in$ $\mathbf{D} \cap \mathbf{Q}$. Otherwise, the dropping arc is (r, s) $\in$ $\mathbf{D}$ such that either $(\frac{1}{\bar{a}_{rs}})\, \beta_{rs}(.)$ = lexicomin $\{(\frac{1}{\bar{a}_{ij}})\, \beta_{ij}(.) : (i, j) \in \mathbf{D} \cap \mathbf{Q}\}$ if (p, q) $\in$ $\mathbf{L}$ or $(\frac{1}{\bar{a}_{rs}})\, \beta_{rs}(.)$ = lexicomax $\{(\frac{1}{\bar{a}_{ij}})\, \beta_{ij}(.) : (i, j) \in \mathbf{D} \cap \mathbf{Q}\}$ if (p, q) $\in$ $\mathbf{U}$.

Notice that, from (2) and (3), $\bar{a}_{ij} \neq 0$ iff (i, j) $\in \mathcal{T}_p \cup \mathcal{T}_q$, and for carrying out the lexicographic rules, we need the rows $(\frac{1}{\bar{a}_{ij}})\, \beta_{ij}(.)$ for each (i, j) $\in$ $\mathbf{D}$. For this reason and since $\mathbf{D}$ is a subset of $\mathcal{T}_p \cup \mathcal{T}_q$, see [2, 6], we define a matrix, called the ratio matrix, whose rows correspond to arcs (i, j) $\in \mathcal{T}_p \cup \mathcal{T}_q$ and whose columns correspond to nodes k $\in \mathcal{N}$. Define

$$R_{ij}(k) = \frac{\beta_{ij}(k)}{\bar{a}_{ij}} \qquad \text{for } (i, j) \in \mathcal{T}_p \cup \mathcal{T}_q,\ k = 1 \text{ to } n \tag{4}$$

and let R denote the matrix ($R_{ij}(k)$: (i, j) $\in \mathcal{T}_p \cup \mathcal{T}_q$ in some order, k = 1 to n). We verify that both $\beta_{ij}(k)$ and $R_{ij}(k)$ are zero for (i, j) $\in (\mathcal{T}_p \cup \mathcal{T}_q)\backslash\mathcal{T}_k$. For each k $\in \mathcal{N}$, let R.(k) denote the column vector ($R_{ij}(k)$: (i, j) $\in \mathcal{T}_p \cup \mathcal{T}_q$). Also $R_{ij}(.)$ denotes the row vector ($R_{ij}(k)$: k = 1 to n). It is straightforward to verify that R.(k) is a zero vector if k is not a node in the quasitrees containing nodes p or q. There are four different cases to consider, these are

Case 1: p $\neq$ q, p and q belong to different quasitrees in the basis network ($\mathcal{N}$, Q).

Case 2: p $\neq$ q, p and q belong to the same tributary tree in a quasitree in ($\mathcal{N}$, Q).

Case 3: p $\neq$ q, p and q belong to different tributary trees in the same quasitree in ($\mathcal{N}$, Q).

Case 4: p = q, (p, q) is a self-loop.

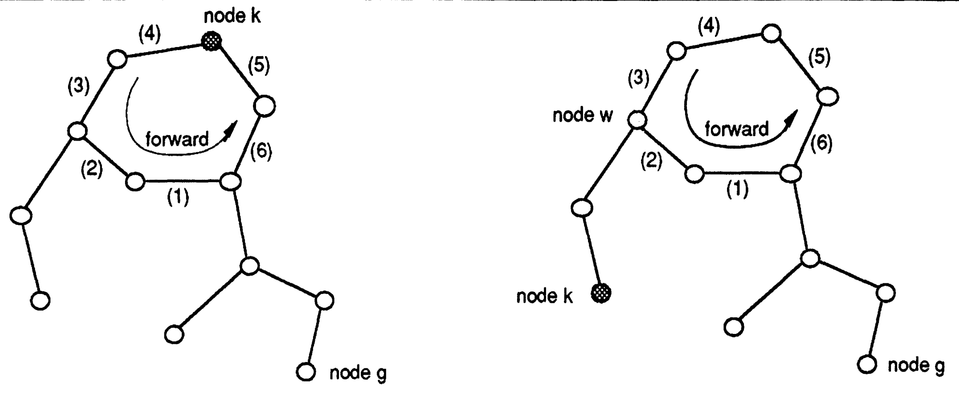
We consider each case separately.

Case 1: $p \neq q$, p and q belong to different quasitrees in the basis network $(\mathbf{N}, \mathbf{Q})$.

Let T_p and T_q refer to the quasitree containing node p and q respectively. We will use T_p and T_q to also denote the set of nodes on these quasitrees. In this case $\bar{a}_{ij} = -\beta_{ij}(p) = -\delta\alpha\psi(\mathcal{P}_{ip})$ if $(i, j) \in \mathcal{P}_p$, where δ, α are defined as in (2) for $k = p$; and $\bar{a}_{ij} = p_{pq}\,\beta_{ij}(q) = p_{pq}\delta\alpha\,\psi(\mathcal{P}_{iq})$, where δ, α are defined as in (2) for $k = q$. From these facts and (4) we have

$$R_{ij}(k) = \begin{cases} 0 & \text{if} \quad k \in \mathbf{N}\backslash(T_p \cup T_q) \text{ and } (i, j) \in (\mathcal{P}_p \cup \mathcal{P}_q) \\ \dfrac{\beta_{ij}(k)}{-\beta_{ij}(p)} & \text{if} \quad k \in T_p \text{ and } (i, j) \in \mathcal{P}_p \\ \dfrac{\beta_{ij}(k)}{p_{pq}\beta_{ij}(q)} & \text{if} \quad k \in T_q \text{ and } (i, j) \in \mathcal{P}_q \text{ and} \end{cases}$$

Let g denote either node p or q and $h_p = -1$, $h_q = p_{pq}$. From above and from (2) we verify that: for $(i, j) \in \mathcal{P}_g$, $k \in \mathcal{P}_g$, $R_{ij}(k) = 0$ if $(i, j) \in \mathcal{P}_g\backslash\mathcal{P}_k$, $R_{ij}(k) = \dfrac{1}{h_g\psi(\mathcal{P}_{kg})}$ if $(i, j) \in \mathcal{P}_k\backslash\mathcal{P}_{kg}$, and $R_{ij}(k) = \dfrac{\psi(C_g)}{h_g\psi(\mathcal{P}_{kg})}$ if $(i, j) \in \mathcal{P}_{kg} \cap C_g$, where C_g is the cycle in the quasitree containing node g. See figure 2(a).



(2a) $k \in \mathcal{P}_g$
Arcs (5) and (6) $\notin \mathcal{P}_{kg}$
Arcs (1), (2), (3) and (4) $\in \mathcal{P}_{kg}$

(2b) $k \notin \mathcal{P}_g$
Arcs (1) and (2) $\in \mathcal{P}_{wg}$
and arcs (3), (4), (5) and (6) $\notin \mathcal{P}_{wg}$

Figure 2: Subcases of Case I resulting of different relative positions of nodes k and g.

For g = p or q and $k \in T_g \backslash \mathcal{P}_g$, define w to be the first common node on $\mathcal{P}_g$ and $\mathcal{P}_k$. So $\mathcal{P}_w = \mathcal{P}_g \cap \mathcal{P}_k$. For $k \in T_g \backslash \mathcal{P}_g$, $R_{ij}(k) = 0$ if $(i, j) \in \mathcal{P}_g \backslash \mathcal{P}_k$, $R_{ij}(k) = \dfrac{\psi(\mathcal{P}_{wk})}{h_g \psi(\mathcal{P}_{wg})}$ if $(i, j) \in \mathcal{P}_w \backslash \mathcal{P}_{wg}$, and $R_{ij}(k) = \dfrac{\psi(C_g)\psi(\mathcal{P}_{wk})}{h_g \psi(\mathcal{P}_{wg})}$ if $(i, j) \in \mathcal{P}_{wg} \cap C_g$. See figure 2(b). Hence we have the following result

Result 4: For g = p or q and $k \in T_g \backslash \mathcal{P}_g$, let w to be the first common node on $\mathcal{P}_g$ and $\mathcal{P}_k$. Then, $R.(k) = \psi(\mathcal{P}_{wk})\, R.(w)$.

Result 5: Let g = p or q. Let $k \in \mathcal{P}_g$, and C_g be the loop in the quasitree of g. If $k \in \mathcal{P}_g \backslash C_g$, $R_{ij}(k)$ has the same value $(= \dfrac{1}{h_g \psi(\mathcal{P}_{kg})})$ for all $(i, j) \in \mathcal{P}_k$. If $k \in C_g$, then $R_{ij}(k)$ has the same value $(= \dfrac{\psi(C_g)}{h_g \psi(\mathcal{P}_{kg})})$ for all $(i, j) \in \mathcal{P}_{kg} \cap C_g$; and a different but the same value $(= \dfrac{1}{h_g \psi(\mathcal{P}_{kg})})$ for all $(i, j) \in C_g \backslash \mathcal{P}_{kg}$. This follows from above.

Using these results, we are now ready to discuss how to implement the lexicographic rules for selecting the dropping arc from the set **D** of eligible dropping arcs in this pivot step. This requires finding the lexicomin or lexicomax among rows $R_{ij}(.)$ of the ratio matrix for $(i, j) \in$ **D**. Recall that all arcs in **D** belong to $\mathcal{P}_p \cup \mathcal{P}_q$. Let **N** be the set of nodes contained in the quasitrees of p or q. For this computation we need to consider only columns of the $(R_{ij}(k))$ matrix for $k \in$ **N**, since $R_{ij}(k) = 0$ whenever $k \notin$ **N**.

The work required is to examine the columns R.(k) for $k \in$ **N** in serial order. In each column we have shown, see Result 5, that there are at most two distinct nonzero entries and, for every $k \in \mathcal{P}_p \cup \mathcal{P}_q$, each of these nonzero entries can be computed by performing a division and a multiplication operations using the collective gain\loss factors of paths along $\mathcal{P}_p$, $\mathcal{P}_q$ that were already obtained as a byproduct during the computation of the pivot column. Thus, to perform the comparison for finding the min or max among rows corresponding to $(i, j) \in$ **D**, in each column R.(k) for $k \in \mathcal{P}_p \cup \mathcal{P}_q$, takes at most a constant amount of work (three multiplications, divisions; and three comparisons). For $k \in \mathcal{N} \setminus (\mathcal{P}_p \cup \mathcal{P}_q)$, let w be the first common node of $\mathcal{P}_k$ with $\mathcal{P}_p \cup \mathcal{P}_q$ as defined earlier. By Result 4, the column R.(k) is $\psi(\mathcal{P}_{wk})$ R.(w) and the comparisons for finding the minimum or maximum in this column among rows corresponding to $(i, j) \in \mathbf{D}$, needs only the sign of $\psi(\mathcal{P}_{wk})$ beyond the entries in R.(w) which were already obtained above since $w \in \mathcal{P}_p \cup \mathcal{P}_q$. For this we define sgn(k), for each $k \in \mathcal{N} \setminus (\mathcal{P}_p \cup \mathcal{P}_q)$, to be the sign of $\psi(\mathcal{P}_{wk})$.

We show that sgn(k) can be computed for all $k \in \mathcal{N} \setminus (\mathcal{P}_p \cup \mathcal{P}_q)$ with a total of at most O(n) effort. First consider the quasitree of p. Let $\mathbf{N}_1$ be the set of nodes in this quasitree not in $\mathcal{P}_p$. For each node $a \in \mathbf{N}_1$ adjacent to $\mathcal{P}_p$ (i.e., there exists an arc (a, d) joining a and a node d on $\mathcal{P}_p$) make sgn(a) = sign of the multiplier of (a, d). Now find a node a_1 in $\mathbf{N}_1$ whose sign is not determined yet, but which is connected by an arc, say (a_1, a_0), to a node $a_0 \in \mathbf{N}_1$, whose sign is already determined. Make $\text{sgn}(a_1) = (\text{sgn}(a_0))$(sign of the multiplier of (a_1, a_0)). Repeat in the same way until $\text{sgn}(a_2)$ is determined for all $a_2 \in \mathbf{N}_1$. Then do the same for the quasitree of q. This whole work requires examining each of the arcs in $(T_p \cup T_q) \setminus (\mathcal{P}_p \cup \mathcal{P}_q)$ once, and hence takes at most O(n) effort.

Case 2: $p \neq q$, p and q belong to the same tributary tree in a quasitree in $(\mathcal{N}, \mathbf{Q})$.

Let T refer to the quasitree containing p and q. We will use T to also denote the set of nodes on this quasitree. As defined earlier (see Figure 1) x here denotes the first common node on $\mathcal{P}_p$ and $\mathcal{P}_q$. Note that x is also a node on the same tributary tree of p and q. Here let g denote one of the three nodes p, q or x. Let $h_p = -1$, $h_q = p_{pq}$, $h_x = -\psi(\mathcal{P}_{xp}) + p_{pq}\,\psi(\mathcal{P}_{xq})$. For each $(i, j) \in \mathcal{P}_p \cup \mathcal{P}_q$, associate the node g to be p if $(i, j) \in \mathcal{P}_{xp}$, q if $(i, j) \in \mathcal{P}_{xq}$, and x if $(i, j) \in \mathcal{P}_x$. Then, as before, for $(i, j) \in \mathcal{P}_p \cup \mathcal{P}_q$, and for $k \in \mathcal{N}$,

$$R_{ij}(k) = \begin{cases} 0 & \text{if} \quad k \in \mathcal{N} \setminus T \\ \dfrac{\beta_{ij}(k)}{h_g \beta_{ij}(g)} & \text{if} \quad k \in T \text{ and } g \text{ is the g-node associated with } (i, j) \end{cases}$$

Note from (3) and (4) that $h_g\beta_{ij}(g) = -\beta_{ij}(p)$ if$(i, j) \in \mathcal{P}_{xp}$, $= p_{pq}\beta_{ij}(q)$ if $(i, j) \in \mathcal{P}_{xq}$, and equals $-\beta_{ij}(p) + p_{pq}\beta_{ij}(q)$ if $(i, j) \in \mathcal{P}_x$. From above and (2), we obtain the following:

$$R_{ij}(k) = \begin{cases} 0 & \text{if} \quad k \in \mathcal{N} \text{ and } (i, j) \in (\mathcal{P}_p \cup \mathcal{P}_q)\backslash\mathcal{P}_k \\ \dfrac{-1}{\psi(\mathcal{P}_{kp})} & \text{if} \quad k \in \mathcal{P}_{xp} \text{ and } (i, j) \in \mathcal{P}_{xk} \\ \dfrac{1}{p_{pq}\psi(\mathcal{P}_{kq})} & \text{if} \quad k \in \mathcal{P}_{xq} \text{ and } (i, j) \in \mathcal{P}_{xk} \\ \dfrac{\psi(\mathcal{P}_{xk})}{h_x} & \text{if} \quad k \in \mathcal{P}_{xp} \cup \mathcal{P}_{xq} \text{ and } (i, j) \in \mathcal{P}_x \\ \dfrac{1}{h_x\psi(\mathcal{P}_{kx})} & \text{if} \quad k \in \mathcal{P}_x \text{ and } (i, j) \in \mathcal{P}_x\backslash\mathcal{P}_{kx} \\ \dfrac{\psi(C)}{h_x\psi(\mathcal{P}_{kx})} & \text{if} \quad k \in \mathcal{P}_x \text{ and } (i, j) \in \mathcal{P}_{kx} \cap C \end{cases}$$

It is straightforward to verify the following result corresponding to Result 4 under Case 1.

Result 6: Let $k \in T\backslash(\mathcal{P}_p \cup \mathcal{P}_q)$. Let w be the first common node on $\mathcal{P}_p \cup \mathcal{P}_q$ and $\mathcal{P}_k$. Then, $R.(k) = \psi(\mathcal{P}_{wk})\, R.(w)$.

From these results we know that the number of distinct nonzero values in the column R.(k) for each $k \in \mathcal{N}$ is at most two in this case. Using this and the same arguments as in case 1, the lexicographic rules can be implemented in this case also with at most O(n) effort.

Case 3: $p \neq q$, p and q belong to different tributary trees in the same quasitree in $(\mathcal{N}, Q)$.

Let T refer to the quasitree containing p and q. We will use T to also denote the set of nodes on this quasitree. Nodes x and y are as defined in Remark 1 (see Figure 1). Note that x (y) is the root node of the tributary tree containing node p (q). Let $h_p = -1$, $h_q = p_{pq}$, $h_x = -\psi(\mathcal{P}_{xp}) + p_{pq}\,\psi(\mathcal{P}_{xq})$, and $h_y = -\psi(\mathcal{P}_{yp}) + p_{pq}\,\psi(\mathcal{P}_{yq})$. From (2), (3) and (4) we obtain:

$$R_{ij}(k) = \begin{cases} 0 & \text{if} \quad k \in \mathcal{N} \text{ and } (i, j) \in (\mathcal{P}_p \cup \mathcal{P}_q) \backslash \mathcal{P}_k \\[2ex] \dfrac{-1}{\psi(\mathcal{P}_{kp})} & \text{if} \quad k \in \mathcal{P}_{xp} \text{ and } (i, j) \in \mathcal{P}_{xk} \\[2ex] \dfrac{1}{p_{pq}\psi(\mathcal{P}_{kq})} & \text{if} \quad k \in \mathcal{P}_{yq} \text{ and } (i, j) \in \mathcal{P}_{yk} \\[2ex] \dfrac{\psi(\mathcal{P}_{xk})}{h_x} & \text{if} \quad k \in \mathcal{P}_{xp} \cup \mathcal{P}_{yq} \text{ and } (i, j) \in \mathcal{P}_{yx} \\[2ex] \dfrac{\psi(\mathcal{P}_{yk})}{h_y} & \text{if} \quad k \in \mathcal{P}_{xp} \cup \mathcal{P}_{yq} \text{ and } (i, j) \in \mathcal{P}_{xy} \\[2ex] \dfrac{1}{h_x\psi(\mathcal{P}_{kx})} & \text{if} \quad k \in \mathcal{P}_{yx} \text{ and } (i, j) \in \mathcal{P}_{yk} \\[2ex] \dfrac{1}{h_y\psi(\mathcal{P}_{ky})} & \text{if} \quad k \in \mathcal{P}_{xy} \text{ and } (i, j) \in \mathcal{P}_{xk} \\[2ex] \dfrac{\psi(C)}{h_x\psi(\mathcal{P}_{kx})} & \text{if} \quad k \in \mathcal{P}_{xy} \text{ and } (i, j) \in \mathcal{P}_{yx} \text{ or} \\ & \qquad k \in \mathcal{P}_{yx} \text{ and } (i, j) \in \mathcal{P}_{kx} \\[2ex] \dfrac{\psi(C)}{h_y\psi(\mathcal{P}_{ky})} & \text{if} \quad k \in \mathcal{P}_{yx} \text{ and } (i, j) \in \mathcal{P}_{xy} \text{ or} \\ & \qquad k \in \mathcal{P}_{xy} \text{ and } (i, j) \in \mathcal{P}_{ky} \end{cases}$$

Again, similarly to case 1, if we obtain the equations for $R_{ij}(k)$ for $k \in T \backslash (\mathcal{P}_p \cup \mathcal{P}_q)$, it is straightforward to verify the following result corresponding to Result 4 under Case 1.

Result 7: Let $k \in T \backslash (\mathcal{P}_p \cup \mathcal{P}_q)$. Let w be the first common node on $\mathcal{P}_p \cup \mathcal{P}_q$ and $\mathcal{P}_k$. Then, $R.(k) = \psi(\mathcal{P}_{wk})\, R.(w)$.

From these results we know that the number of distinct nonzero values in the column R.(k) for each $k \in \mathcal{N}$ is at most three in this case. Using this and the same arguments as in case 1, the lexicographic rules can be implemented in this case also with at mos O(n) effort.

Case 4: p = q, (p, q) is a self-loop

Let T refer to the quasitree containing p and q. We will use T to also denote the set of nodes on this quasitree. In this case $\bar{a}_{ij} = -\beta_{ij}(p)$ if $(i, j) \in \mathcal{P}_p$ and (p, q) is a surplus self-loop; and $\bar{a}_{ij} = \beta_{ij}(p)$ if $(i, j) \in \mathcal{P}_p$ and (p, q) is a slack self-loop. Let $h_p = -1$ or $+1$ depending on whether the entering arc is a surplus or slack self-loop. From these facts and (4),

$$R_{ij}(k) = \begin{cases} 0 & \text{if} \quad k \in \mathcal{N}\backslash T \text{ and } (i, j) \in \mathcal{P}_p \\ \dfrac{\beta_{ij}(k)}{h_p \beta_{ij}(p)} & \text{if} \quad k \in T \text{ and } (i, j) \in \mathcal{P}_p \end{cases}$$

Let C be the loop in T, different from the self-loop (p, q). From above and from (2) we verify that:

$$R_{ij}(k) = \begin{cases} 0 & \text{if} \quad k \in \mathcal{N} \text{ and } (i, j) \in \mathcal{P}_p\backslash\mathcal{P}_k \\ \dfrac{1}{h_p \psi(\mathcal{P}_{kp})} & \text{if} \quad k \in \mathcal{P}_p \text{ and } (i, j) \in \mathcal{P}_k\backslash\mathcal{P}_{kp} \\ \dfrac{\psi(C)}{h_p \psi(\mathcal{P}_{kp})} & \text{if} \quad k \in C \text{ and } (i, j) \in \mathcal{P}_{kp} \cap C \end{cases}$$

Again, it is straightforward to verify the following result corresponding to Result 4 under Case 1.

Result 8: Let $k \in T \backslash \mathcal{P}_p$. Let w be the first common node on $\mathcal{P}_p$ and $\mathcal{P}_k$. Then, $R.(k) = \psi(\mathcal{P}_{wk})\, R.(w)$.

From these results we know that the number of distinct nonzero values in the column R.(k) for each $k \in \mathcal{N}$ is at most two in this case. Using this and the same arguments as in case 1, the lexicographic rules can be implemented in this case also with at most O(n) effort.

5. Conclusion and Extensions

This paper introduces an analytical approach for studying lexicography in generalized network problems. The equations obtained help us to understand and extend the existing theory. First, it is verified that all nonzero entries have the same sign in each row vector of the basis inverse for a generalized network problem with positive multipliers. However, this property does not necessarily hold when there exists negative multipliers. Second, we developed a special implementation of the lexicographic rules for the GN simplex algorithm when addressing GN problems with positive and *negative* multipliers. This strategy evidently avoids cycling and, unlike the SC rules, it requires performing some comparisons. However, the values to be compared are already known since they can be obtained as a by-product of the calculations necessary to compute the basis representation of the entering arc. Consequently, the computational effort per pivot step is O(n) in the worst case analysis. This worst case computational effort is the same as that required by the SC rules for selecting the dropping arc in the method of strong convergence. This is an improvement since classical implementations of the lexicographic rules requires an worst case effort of order $O(n^2)$.

The following extensions will be explored in a subsequent paper. First, due to the special characteristics of the matrix R from Section 3, the comparisons mentioned above can be performed if one knows only the signs of the elements to be compared. Hence, although the actual values are obtained without any additional work, they are not really necessary. Simple rules can be introduced so that the comparisons can be realized by using only the sign information. More efficiency can be expected since comparing boolean variables requires less effort than comparing real ones. Moreover, memory requirements should decrease. Second, the matrix R allow us to develop a constructive proof of the equivalence between the strongly convergent and the lexicographic rules. This approach is more enlightening than existing methods for showing this equivalence.

6. Acknowledgement

This material is based upon work supported by the National Science Foundation under Award Nº EECS-885101 and by the Brazilian Council for the Development of Science and Technology - CNPq under Award Nº 20.2911/85.

References

[1] J. C. Arantes, Resolution of Degeneracy in Generalized Networks and Penalty Methods for Linear Programs, Ph.D. Dissertation, Dept. of Industrial and Operations Engineering, The University of Michigan, Ann Arbor (1991).

[2] J. Elam, F. Glover and D. Klingman, A Strongly Convergent Simplex Algorithm for Generalized Networks, Math. of O. R. 4(1979)39-59.

[3] F. Glover, J. Hultz, D. Klingman and J. Stutz, Generalized Networks: A Fundamental Computer-Based Planning Tool, Mgmt Sci. 24(1978)1209-1220.

[4] F. Glover, D. Klingman and J. Stutz, Extensions of the Augmented Predecessor Index Method to Generalized Network Problems, Transportation Science 7(1973)377-384.

[5] K. G. Murty, *Linear Programming* (John Wiley & Sons, 1983).

[6] K. G. Murty, *Network Programming* (Prentice Hall, 1992, to appear).

[7] J. B. Orlin, On the Simplex Algorithm for Networks and Generalized Networks, Mathematical Programming Study 24(1985)166-178.

[8] M. H. Partovi, A Study of Degeneracy in the Simplex Algorithm for Linear Programming and Network Flow Problems, Ph.D. Dissertation, Dept. of Industrial and Operations Engineering, The University of Michigan, Ann Arbor (1984).